Patent Office
Canberra

I, JONNE YABSLEY, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. 2002951790 for a patent by AMCOR LIMITED as filed on 02 October 2002.

WITNESS my hand this
Thirteenth day of October 2003

J R Yabsley

JONNE YABSLEY
TEAM LEADER EXAMINATION
SUPPORT AND SALES

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION

Applicant:

AMCOR LIMITED

A.C.N. 000 017 372

Invention Title:

UNIFIED DESIGN SPACE

The invention is described in the following statement:

SYSTEM AND METHOD FOR CONTROLLING A DESIGN PROCESS

The present invention relates to a system for controlling a design process which finds particular application in the packaging industry.

Field of the Invention

In many industries the design process consists of a number of sub-design processes which relate to the overall design of a product or process. These individual design sub-processes must be combined to produce an integrated outcome. For example, in the packaging industry in order to deliver a product to the consumers there needs to be a single consistent outcome including the flexible wrap which surrounds the product, the carton in which the flexibly wrapped products are contained for presentation at retail stores and the corrugated box in which the individual cartons are shipped to the retailer. However, the individual elements of this outcome are the result of a number of individual design sub-processes which need to be combined and consistent.

Previous approaches to integrating design sub-processes have generally been linear. That is, an *ad hoc* decision has been reached deciding which design sub-process to begin with and the order in which the design sub-processes will be carried out. As a result, there is limited intelligent and timely interaction between related design sub-processes as well as little to ensure that the outcomes of individual design sub-processes are consistent with the goals of the design process and satisfy some local and/or global optimum requirement. Accordingly, the outcome of the design process is often far from optimum and the design process itself can follow a convoluted path requiring many iterations and accordingly the design process typically takes much longer and requires much more

effort than is desirable.

Further, existing systems make it difficult for different entities to engage within an integrated design space.

In our PCT application, PCT/AU01/00056, we proposed a system which is designed to allow the integration of a number of related software design processes to ensure that the solutions of a first design sub-process module do not conflict with solutions of a second design sub-process.

We now propose a system for controlling a design process which is able to control the manner in which the design process is carried out.

Summary of the Invention

There is provided a system for controlling a design process having a first design sub-process and a second design sub-process, outcomes of one of the first and second design sub-processes being capable of affecting outcomes of the other of the first and second design sub-processes and vice versa because of a relationship between one or more variables (A) of a first design sub-process and one or more variables (B) of a second design sub-process, the system having a user configurable interface between said first and second design sub-processes, said user configurable interface allowing a user of a system to control said design process by specifying which of said one or more variables (A,B) of said first and second design processes are treated as "active" variables and which of said one or more variables (A,B) are treated as "passive" variables, wherein the domain of an "active" variable can be modified by an internal process within the sub-process to which the "active" variable belongs whereas

the domain of a passive variable is predominantly determined by the domains of the other variable or variables in said relationship, whereby specifying which of the variables are treated as passive and which are treated as active determines the manner in which said relationship is evaluated and the dominance of a sub-process in the overall design process.

The internal process may include a user modification of the domain of the active variable.

The domain of a passive variable may be determined directly or indirectly by way of the relationship specified between variables by a rule or an algorithm.

The user configurable interface may define a plurality of relationships between the first and second design processes and said user configurable interface allow a user to select a relationship from the plurality of relationships.

The user configurable interface may allow a user to select a plurality of relationships.

Selection of the relationship may lead to specification of which variables are treated as active and which are treated as passive.

Alternatively or in combination with the above approach, said user may specify a goal or goals of said design process in order to configure said interface. In which case, said user configurable interface specifies said relationship on the basis of the user specified goals.

There may be more than one relationship between

the first and second design sub-processes. For example, between different variables or different sets of variables. Alternatively, one of two or more relationships may be chosen in accordance with a preset or user specified rule. For example, a relationship may need to be changed if the value of a variable increases beyond a preset value. A relationship may consist of one or more rules, one or more algorithms or a combination of rules or algorithms.

Further, there may be more than two sub-processes and accordingly there may be relationships between all or some of the sub-processes as well as relationships between more than two relationships - e.g. either one to many relationships or many to many relationships.

Still further, the system may allow constraints to be placed on a domain of a variable. The constraints may be "hard" constraints which cannot be breached - e.g. to exclude unworkable values of the variable - or "soft" constraints which can be breached in certain circumstances - e.g. if a pre-existing solution of the design process is just outside the constrained domain.

In this respect, in some embodiments, the system has the ability to compare potential solutions to the design process with pre-existing solutions to enable pre-existing solutions to be brought to the attention of a user.

The system typically has an optimisation engine for optimising the design process. Typically, the optimisation engine will use one or more rules to analyse the available solutions. These rules may embody industry experiences or be developed by the system based on the outcomes of previous design processes - i.e. the system may have the capability to learn.

Similar optimisation engines may run within individual design processes.

Brief Description of the Drawings

Figure 1 shows an interface between first and second design sub-processes.

Figure 2 shows an interface between three sub-processes.

Description of the Preferred Embodiment

A design process has the end goal of providing a unified outcome of the design process which can then be put into practice. Complex design processes are typically the combination of a plurality of design sub-processes managed by different entities. Here, the term "entity" is used to designate a person, group of people or organisation or a software module which has responsibility for a design sub-process. That is, the entities are not necessarily different legal entities although they may be - for example, when two entities compete to provide the outcome of a particular sub-process.

It will be appreciated that for a given problem specification each sub-process is capable of producing a wide variety of outcomes but these outcomes must be integrated. Accordingly, it will also be appreciated that in controlling the overall design process there are a number of challenges. These include ensuring that the outcome satisfies the constraints of individual sub-design processes, allowing flexibility in the design process and, as far as possible, ensuring that the outcome is near optimal.

We have recognised that this requires the ability to exercise control over the influence that individual design processes and design variables have on the overall outcome of the design process.

Accordingly, embodiments of the present invention provide a system for controlling a design process which includes a user configurable interface between design sub-processes which allows a user to control the influence individual design sub-processes have on the outcome of the overall design process. The embodiments of the system also allow the design processes to be more flexible than existing design processes and to control collaboration of different design processes.

The user configurable interface is designed to be used by a user who has control of the overall design process. Herein this user is designated as the "project manager" in order to distinguish the project manager from other users of the system - e.g. users of individual sub-processes.

In the preferred embodiment, the user configurable interface is a web (internet or intranet) or client server based application which allows the project manager to control the design process as well as for the individual sub-processes.

The project manager specifies which design sub-process and which variables of individual design sub-processes will form part of the overall design process as well as defining the manner in which they will influence the overall design process. Further, the user configurable interface allows the user to specify the relationships between individual variables which are used in the rules which are used in order to evaluate which outcome or outcomes should be adopted for the design sub-

process. Herein the term "variable" is used in a broad sense to refer to the aspects of a design sub-process which may be varied. In some cases a sub-process will also have "parameters" which are generally unchanging variables and which need to be made available to another process – e.g. because of a relationship with that process.

The relationships between the sub-processes will determine the extent to which sub-processes are dependent on each other - i.e. the extent of coupling of individual design sub-processes to other sub-processes.

The user configurable interface will now be described in relation to a design process which involves first and second sub-processes for simplicity of description. Persons skilled in the art will appreciate that the invention can be extended to any number of sub-processes.

In the web-based application interface a use of a design sub-process is informed of required tasks by an e-mail message or other appropriate messaging system, typically by a workflow process, and with a hyperlink to the system. The user only has access to appropriate pages associated with a set of tasks they are required to perform and if required their task brief. The pages include a set of choice or input variables, which are implicitly constrained. The current constraints (range and or discrete) for variables are given where appropriate and editable if active and not if passive (e.g. visible but greyed out). Alternative techniques could include a portal or an auction process where the participants would be supplied with a specification (containing similar variables and constraints) to satisfy and provide solutions the and associated quotes through the portal which would then be incorporated into the integrated

process.

Setting up the Design Process

In the preferred embodiment, a configuration module is used to configure the interface. The configuration module allows a project manager to specify the goals of the design process. The system then uses embedded logic to determine from the goals which sub-processes and which variables will form part of the interface between sub-processes as well as which rules and algorithms are used to specify the relationships between the variables of the first design sub-process and the variables of the second design sub-process. In this way, the user configurable interface is able to embody user experience. However, the system also allows a user to individual tailor the interface. That is, the project manager may either specify the interface from scratch or may modify the interface which is provided by the system in response to the project manager defining their goals.

Variable, Algorithms and Rules

The three main components which allow the system to be implemented is the manner in which the variables, algorithms, rules and goals are handled.

Variables can be assigned different characteristics which determine the manner in which relationships between variables are evaluated and the dominance of a sub-process in the overall design process. Variables can be assigned either an "active" or "passive" nature.

The domain (i.e. the set of possible values) of a "passive" variable will be determined by other variables, and rules and algorithms. That is, a passive variable

becomes a dependant variable in a relationship with other variables. The relationship is typically specified by one or more algorithms and/or one or more rules. Accordingly, the sub-process to which the variable belongs will respond to changes made to the variable through its relationship with other variables but will not be able to modify its values. However, the sub-process will be able to constrain the domain of the variable so that it does not take unworkable values.

In contrast, an "active" variable can be modified by the sub-process with which it is related again, within constraints set for that variable by the sub-process. Therefore, changes to an active variable by a sub-process will propagate to variables with which the active variable is related by way of the relationship specified by an algorithm and or rule.

A variable may also have "hard" or "soft" constraints.

Typically, a "hard" constraint will be applied by the sub-process. A hard constraint in effect indicates that the sub-process is incapable of providing solutions which lead to that variable taking a value outside a particular range so that if a relationship with another variable calls for the variable to change, there is a mechanism for preventing it from taking an unacceptable value. A soft constraint, on the other hand, could be applied by either the sub-process or the user interface. Therefore, the project manager may apply a soft constraint to indicate the constraint is not significant in the overall scheme of things. For example, a project manager may place a soft constraint on the area or volume efficiency of a packing arrangement or material performance to cost ratio. The latter are typical examples of constraints where there is not a strong degree

of certainty in their choice and hence there may not be a good reason to adhere to the constraint. In addition in some cases they may not have much of an overall affect on an optimisation objective function (e.g. cost), and hence should not be a strong determinant of the solution space (i.e. the possible outcomes), whereas a hard constraint generally has a strong physical limit. During optimisation, if the current solutions do not meet a specified criteria, the optimisation engine may modify the domain of the softly constrained variable to search for better solutions. The soft constraints can prevent excellent solutions being dismissed for spurious reasons - for example because an inappropriate and unrealistic or uncertain choice of efficiency level which may not in fact have a great deal of influence on the cost effectiveness of the solution. The system can be configured so that the type of constraint defaults to soft for some types of variables.

From the perspective of a sub-process the soft constraint may indicate that the sub-process prefers to provide solutions within a specific range but will move outside of that range if necessary.

The rules are embodied by one or more sets of rules engines. The rules engines manage changes and variables against the goals of the design process. The rules engines are a type of software agents and have the ability to incorporate and evaluate rules, rule goals, goal decomposition and assertions and initiate choices and actions based on in-based logic.

Algorithms take a number of different forms. Algorithms may specify a relationship between variables for use in solution generation. Algorithms may also manage the effect of changes to ensure that the constraints of variables are not breached and to ensure

that solutions proposed by one sub-process are allowable by another process. That is, the algorithm propagates constraints and manages variable domains.

Algorithms may also conduct the procedure of searching for solutions, analysing performance of solutions, cost estimation, size determination and manufacturing load.

Example 1

In order to facilitate understanding of the invention, an example of an embodiment involving two sub-processes will be described where the first sub-process (SP1) is a carton and the second sub-process (SP2) is a corrugated box. Figure 1 provides a schematic representation of the components in the system, eg variables, algorithms and rules. Figure 1 shows the connection between variables, algorithms, and rules and the passive and active interface variables. In this example all algorithms and rules may modify or examine an interface variable. Accordingly, in Figures 1 and 2, all lines are shown passing through a central hub 70 to indicate that all rules/algorithms may communicate with all variables. The directed lines (i.e. lines with arrows e.g. line 40) show where a sub-process algorithm or rule may change an active variable. As shown within sub-process 2 where a passive variable cannot be changed by the algorithms and rules of the sub-process, non-directed lines (i.e. lines without arrows, e.g. line 41) show communication/interaction between components. There will also be cases where a rules component does not interact with the interface variables directly but does so via an algorithm.

The project manager may choose one of the following performance goals or sub-goals:

1. The strengths of the carton and box to be determined independently
2. A proportion of the strength of the carton to be used to reduce the strength requirement of the box
3. A proportion of the strength of the box to be used to reduce the strength requirement of the carton
4. A combination of goals 2 and 3 which minimises the integrated carton/box cost.

The project manager selects goal 2, the configuration module configures the interface so that the interface algorithm IA-1 determines a proportion of the carton strength to be used to reduce the required strength of the box during the design process. This goal is appropriate, for example, in a situation where the carton has already been designed or where the primary savings are being sought from the box.

The project manager selects as part of the project configuration (specifically or by a previously configured and named configuration) a series of goals to be satisfied including goal 2 above. The system activates the appropriate algorithms and rules and sets variables as having passive/active constraints and with an associated soft/hard nature. In this embodiment, the system is able to activate algorithms within the sub-process. In other embodiments, particularly where a sub-process is run by a different business, the system may deliver a specification to the sub-process advising it of the required variables and, for example, the strength/performance algorithm which must be used by the sub-process.

In this example, the system itself activates:

- within SP1
  - An algorithm and a rule for strength/performance A1-1 & R1-1, and an algorithm and a rule for constraint management A1-2, R1-2.

- within SP2
  - An algorithm and a rule for strength/performance A2-1 & R2-1, and an algorithm and a rule for constraint management A2-2 & R2-2.
- within the interface
  - An algorithm and a rule for strength/performance interaction IA-1 & IR-1, and an algorithm, IA-2 and rule IR-1 for constraints management.
- Globally
  - A global Constraints Manager & Global Rules Engine maintaining overall control of constraints management, utilising the algorithms and rules and goals such as detailed above e.g. by resolving conflicts between algorithms.

These choices set variables as "active" and "passive" which are in principle straightforward - in setting up a project select the set of goals which implicitly contained such choices within a set of goals. Alternatively, the project manager may choose to engage this directly. The user for each sub process variables will be presented with options and a guide to choices (e.g. a wizard).

Here the sets of internal variables represented within V1 within SP1 are size, board structures, environment, art components and art template. A1-1 is the "selected" set strength/performance algorithm for SP1 and R1-1 the associated rules. For estimates of carton compression loading and/or panel bulge, each performance relationship in A1-1 is associated with one or more styles. An output set from algorithm A1-1 is the maximum allowable load and associated stiffness (i.e. load deflection response). This will typically occur during solution generation, locally and or globally. Before solution generation begins, eg during user interaction, the interface determines whether there are viable

solutions of the design process by determining whether constraints of one sub-process prevent the other sub-process from producing valid solutions. During the variable domain definition phase A1-2 processes/manages constraints and variable domains, including constraint propagation and R1-2 the associated rules. Subsequently, the set of internal variables associated with SP1 are dynamically set/modified by a user or users associated with SP1, which are monitored and processed by A1-2. Amongst other functions, A1-2 sets or validates the domains of variables internally and if possible on the SP1 interface. In addition IA-2 propagates this change to SP2 for processing and validation via the constraint manager in SP2 namely A2-2. Similar comments apply to changes to variables within SP2.

Constraint propagation is handled by an appropriate constraint programming technique. Constraint Programming (CP) is a paradigm for constructing and solving classes of problem in which the domains of the variables involved are constrained and are understood in the context of a constraint domain. The problem solution consists of determining combinations of variable values that are consistent within the domain/constraints. This is termed constraint satisfaction. A Constraint Solver software component determines if the problem is satisfiable, for every constraint. Given a particular constraint domain (Real, FD Finite Domains, Integer, Rational, Boolean, Tree), associated Constraint Solver and Constraint Simplifier defines a language for developing programs and a means of goal(s) evaluation. A constraint domain details the primitive constraints, e.g. $Y \leq aX$, and a constraint is a conjunction of primitive constraints, e.g. $Y \leq aX \wedge Y \geq bZ + cX$. The Solver and Simplifier are used in the evaluation of goals. Such a system typically includes backtracking, bounds consistency for pruning domains, propagation rules and branch and bound solvers

for optimisation problems.

An example of propagation rules used in the pruning of domains, e.g. for $X = Y \times Z$ (for values greater than zero) are:

$$X \geq Y_{min} \times Z_{min},\ X \leq Y_{max} \times Z_{max}$$
$$Y \geq X_{min} / Z_{max},\ Y \leq X_{max} / Z_{min}$$
$$Z \geq X_{min} / Y_{max},\ Z \leq X_{max} / Y_{min}$$

Some typical relationships and interactions associated with domain and constraint management are as follows. The domains of the variables V1 are checked for consistency against the design logic and constraint satisfaction. This is carried out by the rules engine R1-2 and constraint manager A1-2, the domains are modified accordingly and the user who has made the change informed in the event that (and why) the constraints cannot be satisfied. For example if a material cannot be used in a specific environment or with any of the styles listed in the style domain then this material is deleted from the material domain associated with V1 and the SP1 interface. This decision and its dependencies may be stored for future consideration if and when the basis for the decision is no longer valid.

The rules engine is implemented using constraint logic programming. Constraint logic programming (CLP) combines the constraint programming paradigm with a logic programming paradigm. Logic programming (LP) deals with symbolic logic computation and is primarily declarative in nature (what) rather than procedural (how). The former is the focus of the programmer/problem modeller and the latter is essentially facilitated by the system. The modelling paradigm is concerned with facts, rules, goals, queries, predicate logic (first order) etc. The CLP paradigm defines a family of languages. Prolog is an

example wherein a LP programming language is extended to encapsulate CP, by introducing other types of constraints in addition to matching, to give an essentially CLP paradigm. CLP's are made up of rules, which can be recursive. These let the programmer define predicates which are simply user defined constraints or relations. Multiple rules allow a predicate to have more than one possible definition.

CLP essentially separates the problem solving process from the constraint management. Rules engines can direct the problem solving process by making choices of values to be assigned to variables but can also reject these assignments, (e.g. in response to a constraint violation). The constraint management process may reduce variable domain choices, possibly at the direction of a rules engine, via constraint propagation. The constraint management process, using constraint propagation, may return the consistent or inconsistent choices to the rules engine for decision making as well as performing automatic consistency checking as variable value assignments are made by selection/decisions etc. Constraints may be activated or de-activated by a rules engine or a constraints manager. The rules engine may also request the constraints manager to supply a reason for a resultant inconsistency.

For each style there will be a number of art templates appropriate for the type of art objects chosen. For each style (in combination with each material) and set of specific art objects, (size, aspect ratio, positioning, art template rules etc), minimum values for, or combination of, each carton dimension can be determined and used as a constraint. The size domain at the interface is the union of domains associated with each style/material/art object set combination. The latter may be modified by size constraints, which are, for example,

supplied by user, or from manufacturing processes etc. In this way the interface size domain of SP1 is presented to, and for use by, the rules engine IR-2 and the interface constraint manager algorithm IA-2 for propagation to SP2.

The algorithms, A1-1 and A2-1 are respectively concerned with determination of a set of carton and box strength performance measures (e.g. top load, stiffness & bulge). IA-1 is concerned with the strength interaction of the carton and box and monitors and utilises the state of the performance measures active variable set on the respective interfaces. A1-1 interacts with V1 to monitor, read or modify the state of the set of variables V1. This may be via an on change event associated with the variable or via a listening process say if there is a highly distributive nature. A change in V1 (e.g. by a user), may trigger activity within A1-1 which may be subsequently propagated into the interface and hence to other sub-process interface variables etc. A1-1 monitors change in a set of variables in V1 via a "listening" or polling process.

As a result of the selection of the goal, the following interface variables for SP1 are set: materials, styles, arrangement/orientation into the box, mass, performance variables set and size/dimensions, and they are all set to be "active". This is selected by the user, generally in the setting up of a goal but may be directly imposed. Each relationship between variables will dictate the extent of the set of choices the user can make, i.e. at least one must be active otherwise the relationship expresses a fact. The interface variables for SP2 are material, style and size/dimensions and performance variables set. Material and style are active variables and the size/dimensions are passive variables. As a result of a variable being given the property of "passive" or "active", via selection of a set of goals, the ability

of a user, or algorithms, of a sub-process to influence the domain of a variable are either enabled or disabled. In this way the user is able to configure the relative dominance of a design sub-process through the selection of design goals. That is, the algorithms and rules selected in association with the goals dictate the way they interact with the internal and interface variables.

A "passive" or "active" property of an interface variable relates to the ability of the sub-process algorithms and rules to change their domain. The interface algorithms and rules may be able to modify a domain if this is required to pursue and satisfy a goal, irrespective of whether the variable is "active" or "passive", from a sub-process perspective.

After the framework for the design process has been specified and checked for consistency, the design process begins. This can be initiated by a user when the user knows they are in receipt of sufficient information or the system can send a message to the user of a sub-process asking them to engage a design task. A user associated with SP1 makes a change to the carton size within V1. Given that there are relationships/rules/constraints involving styles and size and styles and materials this change activates A1-2, which then reassesses the style, board structure, and size domains relative to the SP1 user's current choice. The choice of art components/art templates will constrain size domains and this impact will be via its effect on the size domain. This results in A1-2 modifying the style, material and size domains on its interface. This interface change then activates constraint manager/algorithm IA-2 (IA-2 monitors change in a set of SP1 interface variables via a "listening" or polling process) which evaluates and propagates the change to SP2, resulting in the size domain of SP2 being modified.

The change to the carton size also triggers the rules engine IR-1 to reassess the strength interaction algorithm IA-1 (the variable change results in IA-1 sending IR-1 a message to re-evaluate the selected set of rules/logic and goals). This can be handled either by IA-1 monitoring variable changes and then passing them to IR-1 or by IR-1 monitoring them directly depending on the complexity of the evaluation. This is a system design decision on how rules are best handled. IR-1 evaluates the state/values on interfaces from SP1 & SP2 and relays back to IA-1 appropriate measures to be taken by or modifications of IA-1 in performance evaluations during the solution generation phase. For example the change activates IR-1. IR-1 evaluates the size domains on SP1 & SP2 and the orientation & arrangement from SP1 (this includes loading allowance and headspace). The result is that IR-1 activates a "bulge performance measure" to be included within IA-1 - given a specific orientation and level of loading allowance. From this analysis the addition of bulge results IA-1 modifying the size domain on the SP2 interface. R2-1 responds to this change and R2-1 takes the new size domain and modifies the style and material domains of SP2.

Each change must be sanctioned. If the variable change associated with SP1 is in conflict with variable domains of SP2 then the latter may be modified (typically expanded) by the interface components, IA-1 or IR-1, within the absolute constraints set by SP2. If the changes cannot be satisfied by modification other than violating latter the user is informed or blocked from making the change.

Assuming the variable domains are in a satisfactory state (i.e. the domain is not empty/null, the constraints are satisfied and solutions can be generated)

the user may initiate an integrated solution generation process over the two design sub-processes (note that, when the design process allows, solution generation or optimisation algorithms local to a sub-process may also be initiated). The constraint manager search algorithm A1-2, IA-2 and A2-2 process and select solution options from the appropriate variable domains. The number of combinations to be tested can be reduced by techniques such as a bounds consistency checking and branch and bound. This process involves determining a carton performance interaction and box performance evaluation. Algorithms IA-1 & A2-1 are used to evaluate the box materials on the basis that the strength requirement can be reduced using the calculated strength contribution and stiffness from the cartons contained by a box. The carton and box specifications, associated with a solution are checked by the appropriate rules engines to ensure suitability of manufacture and compatibility with filling lines. Given that an integrated outcome is consistent with the variable domains/constraints and rules it is saved to memory and subsequently presented to the project manager as part of a solution set associated with the current state of the design definition.

Where solutions do not present themselves, the constraint manager may perform checks against soft constraints. Constraint manager typically embodies "fuzzy logic" to enable it to determine wherein self constraint should be breached.

The project manager can then select a solution or alternatively ask for further analysis to be performed (e.g. on cost or setting another goal to differentiate solutions) before selecting a solution.

Example 2

Referring now to Figure 2 there is shown a schematic representation of an interface between three sub-processes. The second example is for illustrative purposes of potential functions and does not relate to specific sub-processes. Box 50 indicates the user interface configuration module which is used by the project manager to configure the design process as indicated by arrows 54. In this embodiment the project manager is capable of configuring first design sub-process SP1, second design sub-process SP2 and third design sub-process SP3 as well as configuring interface 55.

In Figure 2, dotted circles represent active interface variables, circles with thick borders represent passive interface variables and circles with thin borders represent internal variables. The letter A is used to designate algorithms, the letter R is used to designate rules and the letter V a set of variables. The numeral to the left of the hyphen indicates to which sub-process the algorithm or rule belongs, and the numeral to the right of the hyphen indicates the number of the algorithm or rule within the module. The letter I indicates that the algorithm or rule belongs to the interface.

As shown by design sub-process 1, a design sub-process may incorporate a number of different algorithms which interact with the variable set under the control or management of a rules engine R1-1. The set of variables 56 are all active and hence tend to drive the overall design process. However, the algorithms A1-1 and A1-2 also interact with other passive variables related by filling line 57 and manufacturing side 58 which cannot be altered by A1-1 or A1-2.

Design sub-process SP3 shows that complex processes may be at work within in a design sub-process with a number of different algorithms and rules competing/combining to produce the result. Further, as indicated by active variable 58, the system can incorporate other active variables which are not part of the design process as far as the interface is concerned. Active interface variable 58 may indicate a separate interface with service provider 59 who runs manufacturing site 60.

The group of algorithms and rules IA-1, IR-1 and IA-2 indicate that for example rules engine 1 may monitor IA-1 and substitute IA-2 for IA-1 if certain predetermined conditions are met. IA-3 indicates that interface algorithms do not need to interact directly with interface variables. IA-3 can perform some intermediate task such as calculating a value from the passive variable with which it is associated and then passing this value to IA-1.

Various modifications will be apparent to persons skilled in the art and should be considered as falling within the scope of the present invention.

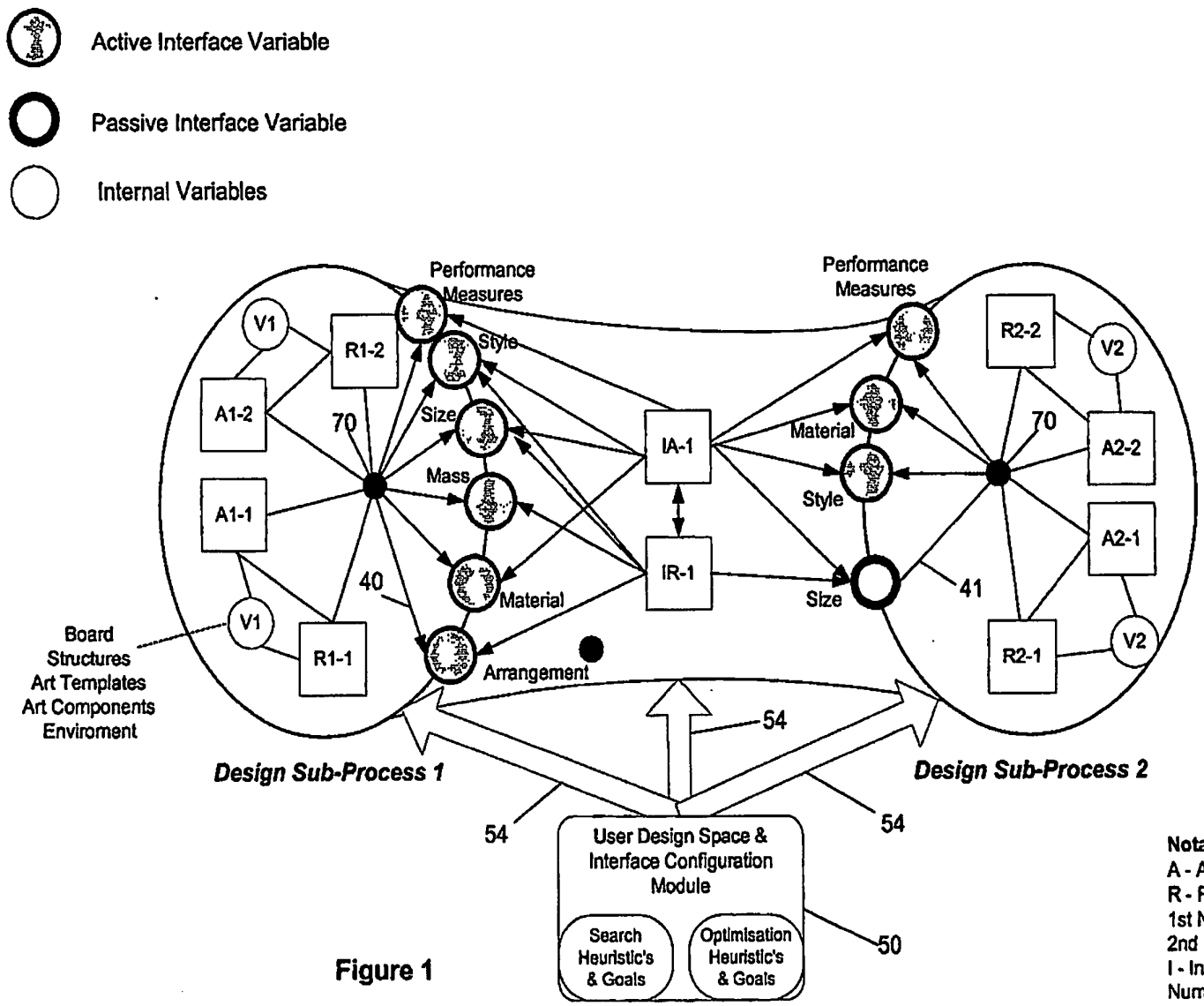

Figure 1

Notation
A - Algorithm
R - Rule
1st Numeral - Sub-Process
2nd Numeral - A/R number
I - Interface
Numeral - A/R number

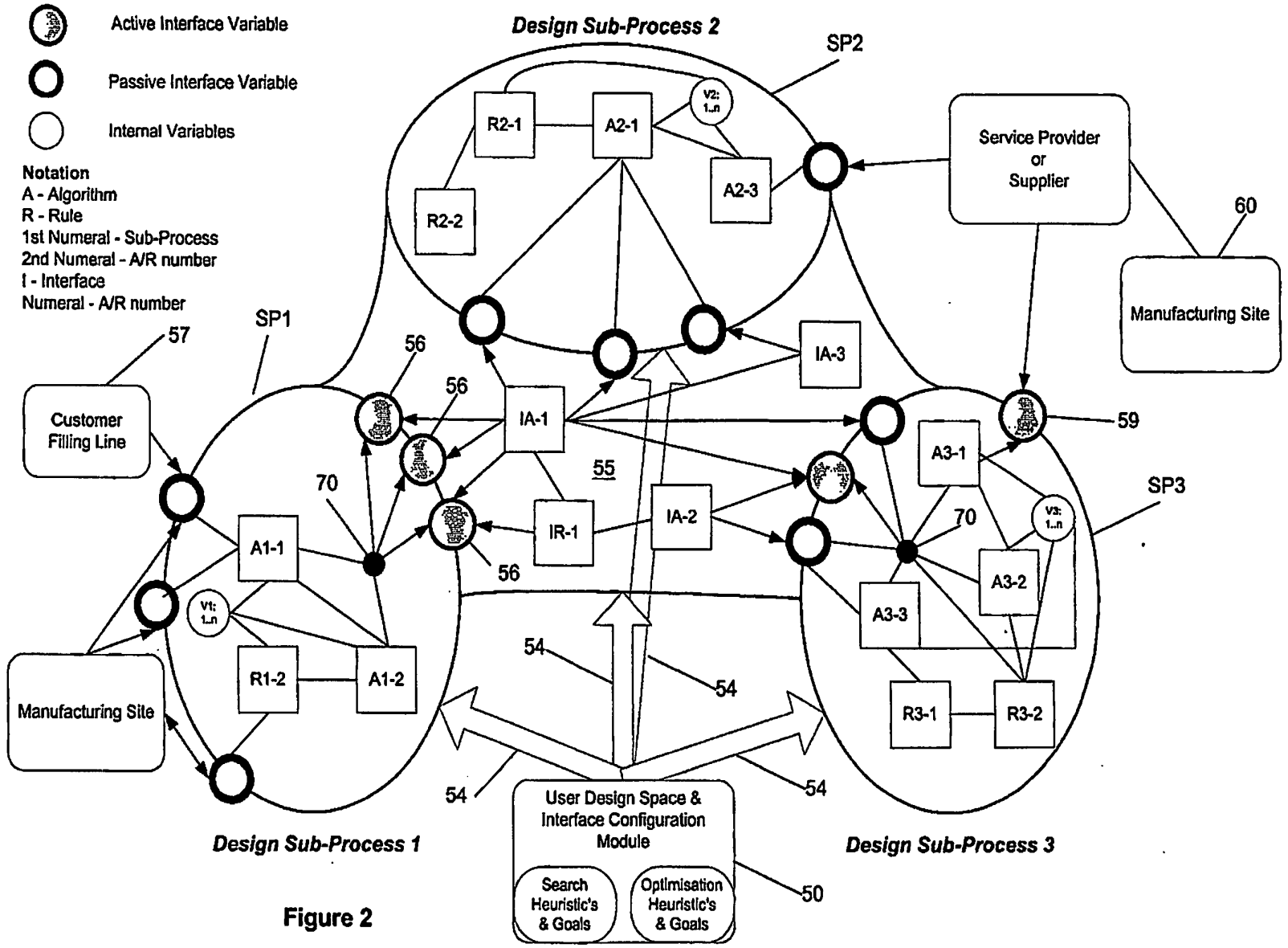
Active Interface Variable
Passive Interface Variable
Internal Variables
Notation
A - Algorithm
R - Rule
1st Numeral - Sub-Process
2nd Numeral - A/R number
I - Interface
Numeral - A/R number
Design Sub-Process 2
SP2
R2-1
A2-1
V2: 1..n
A2-3
R2-2
Service Provider or Supplier
60
Manufacturing Site
SP1
57
56
56
IA-3
IA-1
Customer Filling Line
59
A3-1
55
70
SP3
IR-1
IA-2
70
V3: 1..n
A1-1
56
A3-2
V1: 1..n
A3-3
54
54
R1-2
A1-2
R3-1
R3-2
Manufacturing Site
54
54
User Design Space & Interface Configuration Module
Search Heuristic's & Goals
Optimisation Heuristic's & Goals
50
Design Sub-Process 1
Design Sub-Process 3

Figure 2